ACEEE Int. J. on Communications, Vol. 4, No. 1, July 2013

Packet Loss and Overlay Size Aware Broadcast in the Kademlia P2P System

Zoltán Czirkos, György Bognár and Gábor Hosszú
Budapest University of Technology and Economics, Department of Electron Devices, Budapest, Hungary
Email: {czirkos,bognar,hosszu}@eet.bme.hu

***Abstract*—Kademlia is a structured peer-to-peer (P2P) application level network, which implements a distributed hash table (DHT). Its key-value storage and lookup service is made efficient and reliable by its well-designed binary tree topology and dense mesh of connections between participant nodes. While it can carry out data storage and retrieval in logarithmic time if the key assigned to the value in question is precisely known, no complex queries of any kind are supported. In this article a broadcast algorithm for the Kademlia network is presented, which can be used to implement such queries. The replication scheme utilized is compatible with the lookup algorithm of Kademlia, and it uses the same routing tables. The reliability (coverage) of the algorithm is increased by assigning the responsibility of disseminating the broadcast message to many nodes at the same time. The article presents a model validated with simulation as well. The model can be used by nodes at runtime to calculate the required level of replication for any desired level of coverage. This calculation can take node churn, packet loss ratio and the size of the overlay into account.**

***Index Terms*—peer-to-peer, distributed hash table, Kademlia, broadcast**

I. Introduction

Broadcast messaging is infrequently used by applications built on peer-to-peer overlay networks. Due to the significant number of nodes of these overlays, broadcast messages generate large network traffic. Structured P2P overlays were, in turn, developed to reduce the traffic generated by data lookup requests, which were based on flooding mechanisms in earlier overlays like Gnutella.

However, some applications still require broadcast messaging. For example, it can be used to implement partial keyword searches, or so called blind searches. Broadcast messages can also be used in any structured peer-to-peer network to disseminate information valuable for all peers. Types of such pieces of information include system-wide configuration parameters or global messages in instant messaging services. Blind search requests are also not much different from broadcast messages: the search requests are forwarded like a broadcast message, but forwarding is stopped once the request is able to be fulfilled.

The topologies of structured overlay networks are organized in such a way, that the diameter of the overlay, i.e. the number of hops between any two nodes in the topology is small. By following this topology, broadcast messages can be delivered to nodes quickly, usually in logarithmic time. Problems arise however when nodes or network connections fail, or malicious (Byzantine) nodes join the network. These hinder the correct functioning of communication, and even cause that most nodes do not receive the message at all.

In this paper, we present a broadcast algorithm for the Kademlia overlay network, which uses replication to ensure reliability. We deduce formulae to estimate the correctness of the algorithm in the function of network packet loss ratio and node reliability, and also to select the required replication factor to overcome these. The results are validated with simulation as well.

II. Related Work

Peer-to-peer (P2P) systems work on application level overlay networks, in which all nodes are clients and servers at the same time [1]. Nodes can use services of others, for which they also allow their own resources by anyone. These resources include shared files, storage space, computational power and such. As connections on the application level are freely chosen by nodes, different topologies can be built, so the most feasible topology can be chosen for a specific application.

A. Structured P2P Networks

Structured P2P networks utilize a predefined topology, in which the *diameter* of the *overlay* network is usually only *logarithmically proportional* to the number of nodes. This makes overlays scalable. To achieve this, different topologies are to be used. *Chord*, for instance, organizes its nodes to a ring using their NodeIDs. Each node has its own connections to neighboring ones, but there are also auxiliary connections to nodes being a half, a quarter, one eighth etc. further away on the ring. Distance between any two nodes is measured on the perimeter of the ring.

Kademlia, in turn, organizes its nodes to a binary tree. (For an in-depth discussion of the internal mechanisms of Kademlia, please refer to [2].) Distance between nodes is calculated using the XOR (exclusive or) function, which essentially captures the idea of the binary tree topology. For any nodes A and B, the magnitude of their distance d(A,B)=$A \otimes B$, e.g. the most significant nonzero bit of d is the height of the smallest subtree containing both of them. The application level routing tables of Kademlia nodes, which store <*NodeID;IP address*> associations, are aligned to these subtrees. The size of the tables is limited to k for any subtree, so for the distant half subtree a node knows at most k nodes, for the distant quarter subtree at most k, and so on. These tables are usually called k-buckets. Lookup requests travel

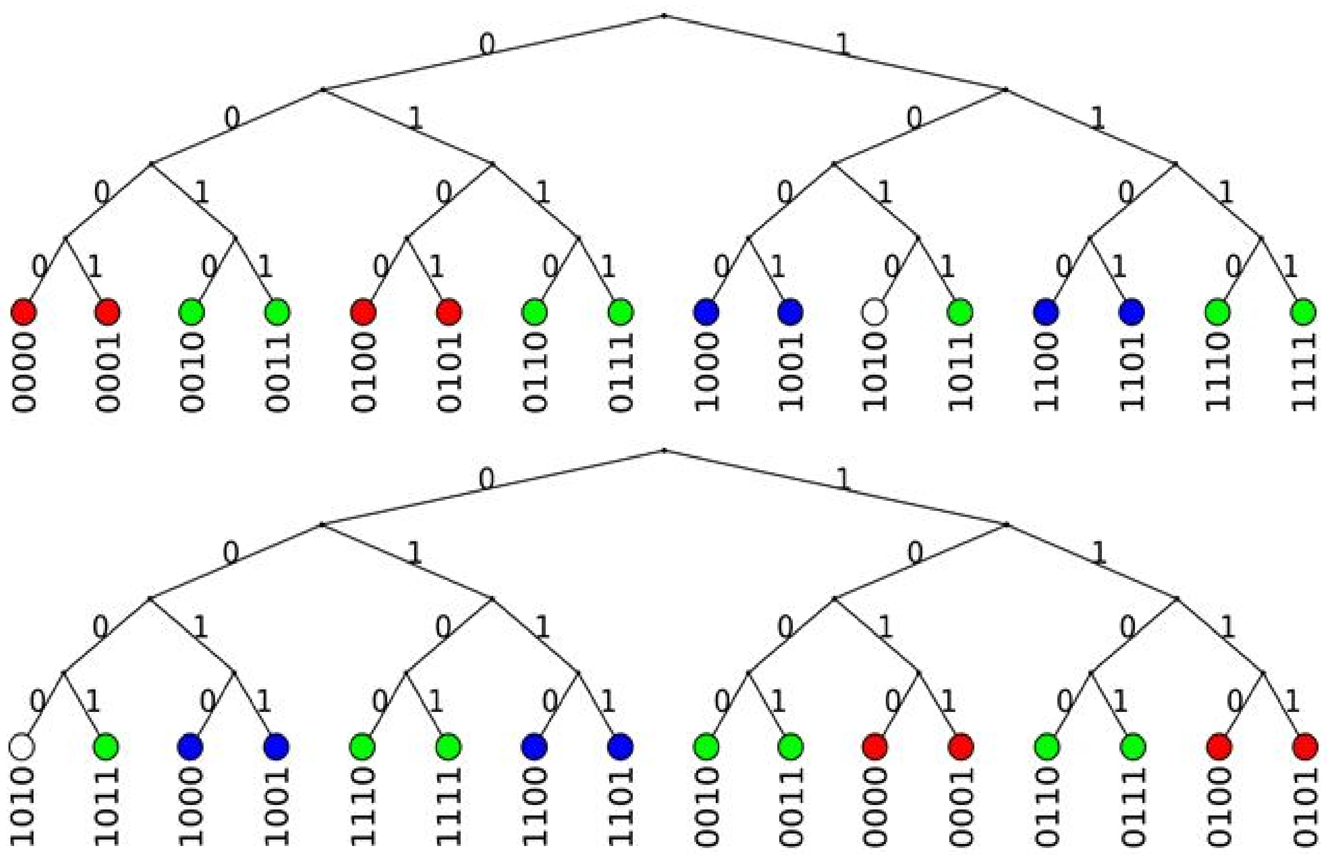


Figure 1. Transformation of the Kademlia overlay. During the transformation, the distances measured using the XOR metric are unchanged, therefore the routing tables of the transformed overlay are the same as those of the original overlay

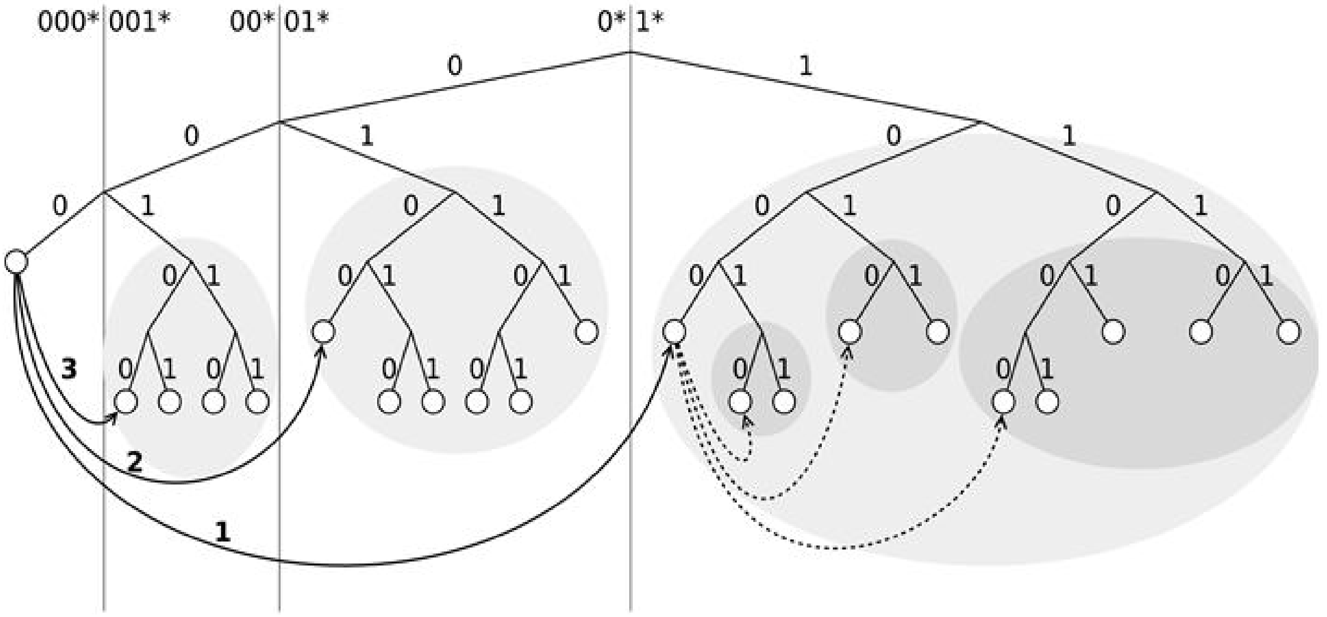


Figure 2. Illustration of broadcast messaging in the Kademlia overlay. The messages sent, besides containing the payload, are selecting responsible nodes for successively smaller subtrees as well

along the path set by decreasing distance, e.g. to smaller subtrees closer in every step to the destination. Any node can lookup an arbitrarily selected one in O(log *N)* steps, where *N* is the size of the overlay. For the lookup procedure to work correctly, at least one node is to be known for all the subtrees of decreasing size. Knowing more nodes enhances stability and provides the ability to initiate parallel lookups for increased speed.

B. Broadcast in P2P Networks

The broadcast algorithms of P2P networks generally use the connections already set by their topology. As discussed above, these already have the property of making the diameter of the network small, and therefore provide short paths for messages sent. By using these, it is not necessary to initiate extra lookup requests to build new paths for dissemination of the broadcast message [3].

Several broadcast algorithms presented in the literature are based on selecting *responsible nodes for specific intervals* of NodeID space. By properly partitioning the address space, the broadcast using *N-1* messages can be sent in O(log *N*) time, where *N* is the size of the overlay. If the tree formed by the path of the messages is higher than that, the broadcast will be slower [4]. The algorithms presented handle churn (nodes quitting, failing or maybe joining) with various levels of reliability and extra cost [5].

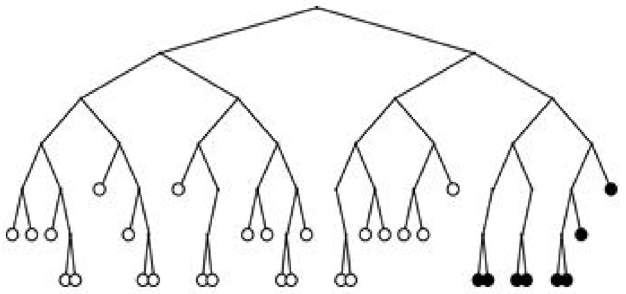


Figure 3. Error of broadcast messaging as it is seen in simulation. If messages sent to reliable peers are lost, complete subtrees are left out of the broadcast

The *Efficient Broadcast* algorithm partitions the address space of Chord into equally sized intervals [6]. For example, in an overlay, which uses the address space [0,8], the node with the address 0 initiating the broadcast will divide the address space into the intervals [0,4) and [4,8). In the latter one, node 4 is responsible for sending the message, while in the former, node 0 is responsible. So it refines the interval to [0,2) and [2,4) again, and node 4 does the same for its own selected interval. The reliability of this algorithm can drop rapidly [3] when it experiences packet losses or incorrect routing tables due to churn [7].

Reliability can be improved by instructing selected responsible nodes to provide an acknowledgement of successful broadcast [8]. This is a recursive mechanism: a responsible node will send the acknowledgement to back to the one it is selected by, if it has received *all* the acknowledgements from the nodes it selected for its smaller intervals. After a predefined timeout, new responsible nodes are selected. While this solution is applicable for routing table deficiencies, it fails to provide adequate correctness when malicious nodes join the overlay, which reply to broadcast requests with false positive acknowledgements.

Other algorithms use *epidemic-style* communication to cover the address space of overlays. By the combination of the two approaches, the number of messages does not increase significantly [3]. For example, in the algorithm *Efficient Broadcast in P2P Grids*, responsible nodes randomly select more neighbors for sending them the message as a duplicate [9]. This ensures reliability in case of nodes failing or quitting the overlay. Algorithms based on selecting responsible nodes are generally faster that the ones working with random selection. However, the latter are more reliable when network errors are present.

Efficiency can be improved if the aim of the broadcast is to provide *blind search*. For these, forwarding the message to neighbors is stopped if the search request is fulfilled [10], or the flooding can also be stopped by assigning a TTL (time to live) field to messages and decrementing it with every hop [11].

III. The Replication-Based Broadcast Algorithm For Kademlia

In this section, the novel broadcast algorithm is presented, which can be parameterized with the level of replication. The simplified operation of the algorithm without replication is presented first, and then reliability is estimated using this model.

A. The Transformation of the Kademlia Address Space

The broadcast algorithm described in this paper assumes that the node initiating the broadcast has NodeID 0. However, algorithms which have this assumption can work on any overlay and can be run by any node, as discussed below.

In Kademlia, any node can be transformed into the 0 point of the NodeID space, and during this transformation thedistance and neighbor relations of the overlay nodes are invariable. The transformation is achieved by using the XOR function on the NodeIDs of the overlay for each of the nodes with a selected transformation constant denoted with *X*. Every nonzero bits of *X* will make specific subtrees of the topology switch spaces. If *X* is 1000…, e.g. the most significant bit is 1, the left and right half subtrees will switch places in the overlay, but the subtrees themselves will be left unaltered. If *X* is 0100…, the quarter subtrees will do the same in both the left and the right half subtree and so on. With the transformation every node is assigned a new NodeID, which will be

$$N' = N \otimes X \qquad (1)$$

However, due to the properties of the exclusive or function, distances of node are unchanged:

$$D = A' \otimes B' = (A \otimes X) \otimes (B \otimes X) = A \otimes B \qquad (2)$$

This implies that the routing tables of the transformed overlay are essentially the same as those of the original, as they are dependent on the distances of nodes in the XOR topology, which are the same in both. To move any node into the 0 point, the NodeID in question has to be used as the transformation constant *X*. Figure 1 shows this operation for node 1010.

B. Broadcast on the Kademlia Topology

The algorithm presented here works similarly as those that operate on Chord as discussed in Section II, but adapted to Kademlia. In this algorithm, the node initiating the broadcast selects nodes from distant subtrees, each smaller than the previous, and sends them the message, as seen on Figure 2. We call these nodes *responsible* or *delegate* for a subtree, because they are responsible for forwarding the

DOI: 01.IJCOM.4.1.1127

broadcast in their own subtrees.

The broadcast uses the *k*-buckets of Kademlia. The node initiating the broadcast will send a message to any node from the distant half tree, which will be responsible for carrying out the broadcast in its own subtree. The initiator node is in turn responsible for its own half, so its sends the message to a randomly selected node from its own half subtree, but its distant quarter tree, and so on. This is continued until the subtrees are run out: in every step, the responsibility for the distant subtree is delegated to a randomly selected node, while the responsibility for its own subtree is kept by the node running Algorithm 1. (Note that bucket[i] in the algorithm refers to the nodes' own *k*-buckets, as those are always sorted by the distances, not the absolute value of addresses.) An important tag in the message for this algorithm is the height variable, which determines the size of the subtree the node is responsible for. This is incremented on each forwarding step. The broadcast is initiated with the *height* value set to zero, e.g. the initiator node is at first responsible for the whole tree.

```
broadcast(message text, height)

for (i = height to number of address bits)
    if (bucket[i] ≠0)
        node ← random node from bucket[i]
        send message to node: message text, i+1
    end if
end for
```

Algorithm 1. Broadcast for Kademlia (without replication)

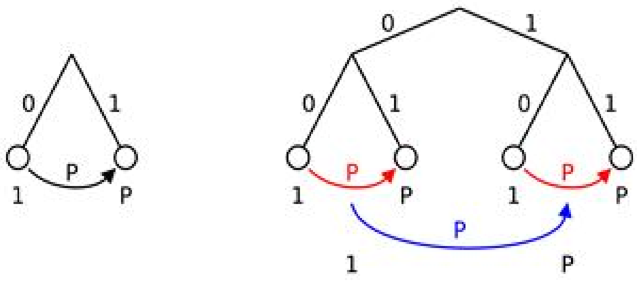


Figure 4. The probability of nodes getting the message for a tree of two and four overlay nodes

C. Reliability Problems Caused by Network Errors

A relevant weakness of the algorithm presented above is that messages sent have significantly different importance, and the level of responsibility delegated also varies much. The delivery of the message for even half of the tree is dependent on the successful delivery of a *single* message. Figure 3 shows a result of a simulation in which a message with *i=2* was lost during transport, causing 25% of the nodes to be skipped. This problem is due to nodes quitting, failing or out of date routing tables caused by churn.

Other figures of merit of the algorithm are affected by the different importance of messages as well. For example, the latency of a message selecting a responsible node for a half tree will have an impact on the broadcast delivery time for the whole subtree it is responsible for. Therefore, the deviation of delivery times for messages is very high.

D. Calculating the Reliability of the Algorithm

The *reliability* of the algorithm, e.g. the *ratio of nodes getting the message* can be calculated by examining the probability of proper responsibility delegation, as seen on Figure 4. The responsibility is *properly delegated* if the message selecting the delegate node is neither lost, nor is the node Byzantine. In the following calculation, P_h denotes the probability of failure. We assume that network packet losses are unrelated, and are therefore uniformly distributed. The probability of successful delegate selection is $P=1-P_h$. The expected *number of nodes* getting the message for two immediate neighboring nodes is $1+P$ accordingly, as seen on the left hand side subfigure on Figure 4. The node *sending* the message has already received it, so its expected value is 1. The node *receiving* this message adds an expected value of $1{\cdot}P=P$.

For an overlay of four nodes, e.g. twice as large, the same argument applies to subtrees. The left hand side subtree will receive the message, as the original initiator is responsible for that. So the expected number of nodes receiving the message in that subtree is $1+P$. On the other hand, in the right hand side subtree this only applies if the responsible node for the subtree receives the message. This also has the probability of P. The expected value for that subtree is $P(1+P)$. Summing these we get $(1+P)+P(1+P)$, which equals to $(1+P)^2$. By generalizing this reasoning, the expected number of nodes receiving the message for a tree of height b is $M=(1+P)^b$. Dividing this by the number of all nodes, 2^b to get the ratio of nodes receiving the message, the result is:

$$m = \left(\frac{1+P}{2}\right)^b \tag{3}$$

Byzantine nodes, which do not forward the message, will have the exact same effect on reliability as lost packets. The algorithm chooses responsible nodes randomly from *k*-buckets, so there are no points of address space which would have any more significance than others. If the overlay has N_h Byzantine nodes, the probability of choosing a Byzantine node to be responsible is $P_h = N_h/N$ with N being the number of all nodes in the overlay. In order for them to be effective, they must have resources comparable to that of the overlay as a whole, which is highly unlikely.

The above reasoning assumes that Byzantine nodes are distributed evenly (or they distribute themselves evenly) in overlay address space. As responsible nodes are selected randomly, there are no intervals in the address space, which have a significant importance for the broadcast. If the Byzantine nodes were aiming to hinder the whole broadcast mechanism in the overlay, they would have to distribute themselves evenly. The more Byzantine nodes are inserted into an overlay, more harm is caused. If these nodes are under control of a specific attacker, he must have resources comparable to that of all nodes in the overlay to cause significant harm, because to increase the failure of a delegate selection to P_h, it has to insert P_h=N_h/N nodes.

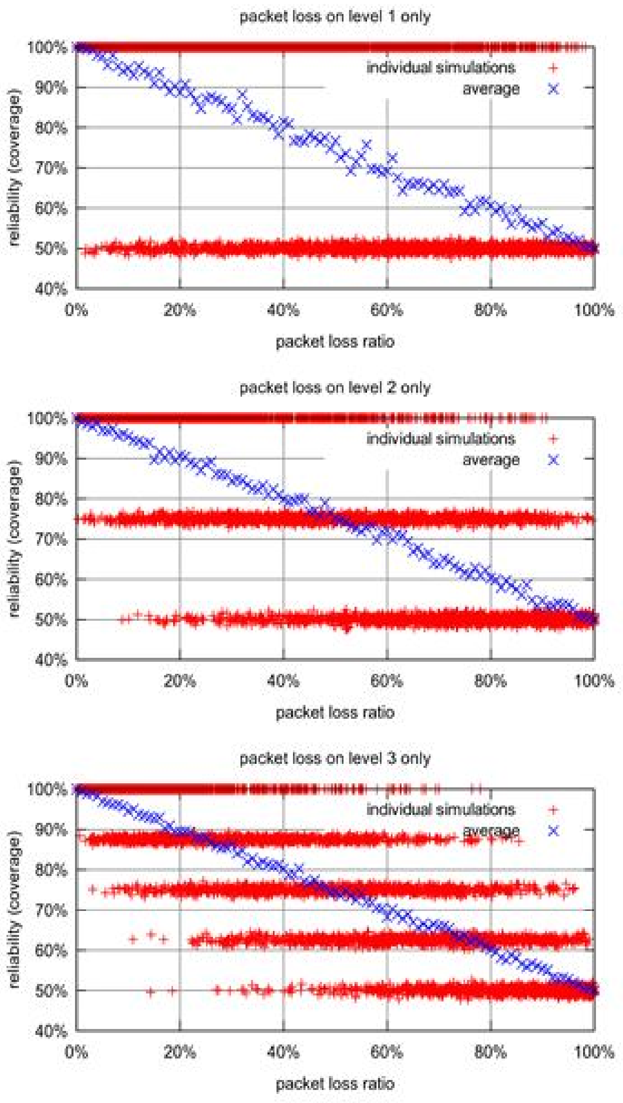


Figure 5. Simulation results for different packet loss ratios occurring on different subtree sizes during the broadcast

E. Broadcast with Replication

To overcome the effect of packet losses and Byzantine nodes, one can use replication. In the upgraded Algorithm 2, more responsible nodes are selected for each subtree. The number of these is denoted with k_b. Applicable values range from 2 to *k*, the size of *k*-buckets. (k_bd”*k* must hold, otherwise one would need extra FIND_NODE requests to get IP addresses of already known nodes for a subtree.) As a node can receive a message more than once, messages are to be tagged with a unique identifier as well.

By selecting more responsible nodes from each subtree, the probability of the message getting lost will decrease. As any responsible node can handle the task of broadcasting for any given subtree, this probability will be $P_h^{k,b}$, where P_h is the probability of a single message getting lost, and k_b is the number of responsible nodes in each tree. The probability of a subtree receiving the broadcast is therefore increased to

```
broadcast(identifier, message text, height)

if (identifier ∈ seen messages)
  return
end if
seen messages ← seen messages ∪ {identifier}
for (i = height to number of address bits)
  if (bucket[i] ≠ 0)
    nodes ← k_b random nodes from bucket[i]
    for all (node ∈ nodes)
      send message to node: identifier, message text, i+1
    end for
  end if
end for
```

Algorithm 2. Broadcast for Kademlia with replication

1-$P_h^{k,b}$. For example, in an overlay with a fairly high ratio of network errors, P_h=10%, selecting k_b=2 will reduce the probability of losing the message to P_h^2=1%, thus reliability is increased from P=90% to P=99%.

Reliability can be increased to any desired level with a properly selected k_b value. By using P=1-$P_h^{k,b}$ and eq. (3), we get

$$m = \left(\frac{2 - P_h^{k_b}}{2}\right)^b \qquad (4)$$

which can then be solved for k_b (rounded up, as replication cannot be a fraction, but only an integer):

$$k_b = \left\lceil \frac{\ln\left(2\left(1-\sqrt[b]{m}\right)\right)}{\ln P_h} \right\rceil \qquad (5)$$

Here *n* is the required level of reliability, P_h is the packet loss (or Byzantine node) ratio, and *b* is the number of address bits which is estimated by b=log$_2N$ for an overlay with *N* nodes.

The required replication level calculated in eq. (5) is only virtually a function of the number of bits in NodeIDs. Real overlay networks never have fully populated address spaces, but much less nodes. (Kademlia uses 160-bit addresses [2], which would mean 2^{160} nodes.) The reasonable size for an overlay is only 10^5 or 10^6 nodes. This implies that the address space of such is essentially empty. The number of hops between any two nodes is therefore much less than *b*. Assuming that nodes are using the address space uniformly (which is the likely scenario, either because their addresses are generated using a proper pseudo random number generator, or calculated using hash functions from their IP addresses), an equivalent value of *b* to be substituted into eq. (4) can be calculated using b=log$_2N$.

Nodes can determine the size of the overlay approximately by examining the density of nodes around a selected location of address space (maybe even their own), as discussed in [12]. The packet loss ratio can also be measured by considering acknowledgement messages to any overlay proto col message. This implies that *eq. (5) can be parameterized by any node* at runtime.

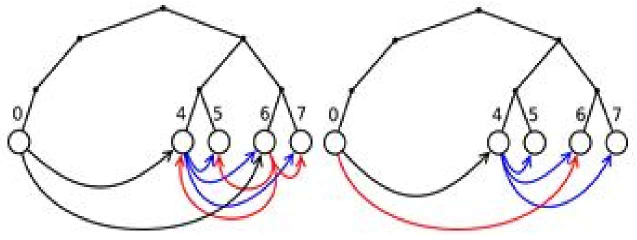


Figure 6. For the discussion of network traffic generated by the algorithm for different latency distributions

F. Differentiated Replication Levels

The different failure ratios caused by lost messages for different trees suggest that replication levels should be handled differently for each subtree size. Intuition suggests that for tall subtrees with many nodes, a higher replication level might be feasible, as a packet loss in these trees cause a greater number of nodes missing the message. However, this is not exactly the case, as replication levels used at different heights of the tree have the same effect on coverage. This can be proved by simulation, and also analytically.

Consider Figure 4 again. The node pair on the left hand side of this figure has an expected value of *1+P* for coverage (as discussed above), whilst the overlay of four nodes on the right hand side part of the figure has an expected value of *1(1+P)+P(1+P)=(1+P)(1+P)*, where P is the probability of proper delegation. On one hand, if we assume the arrival of messages in the small subtrees (red arrows), we get either 50% or 100% coverage for the full tree, depending on the arrival of the message between the two trees (blue arrow). On the other hand, by assuming the arrival of the blue message, the coverage of the tree can be 100% (if both red messages arrive), 75% (if one of them is lost) or 50% (if both get lost during transport). This means that coverage can still drop to 50%, because success depends on the two messages, not only one.

By considering a subtree of any size, we can observe the same phenomenon. The possible coverage values will be different (more subtree sizes will appear), but the expected value of coverage will be independent of the size of the subtree the packet loss occurs in. Proper delivery will depend on more and more messages with the increasing tree size. Figure 5 shows this in simulation. The red dots illustrate individual simulations, and the blue dots show the average value (expected tree coverage) for the packet loss ratio selected by the x axis. As we increase the subtree size, on which the packet loss occurs, the expected value function remains the same. (The small fluctuation of individual coverage values around 75%, 50% etc. is caused by the imbalance of the overlay trees simulated.) The overlays in these simulations contained 4096 nodes.

This means that in order to achieve a desired level of coverage, the expected value of proper delegation must be kept high in all subtree sizes. This measure is where P_x is the probability of proper delegation in level x. By substituting

$$\mathrm{m} = \frac{1+P_1}{2}\cdot\frac{1+P_2}{2}\cdot\frac{1+P_3}{2}\cdot\ldots, \qquad (6)$$

$P=1-P_h^{k,b}$ with differentiated k_b replication levels for each subtree, we get:

$$\mathrm{m} = \frac{2-P_h^{k_{b,1}}}{2}\cdot\frac{2-P_h^{k_{b,2}}}{2}\cdot\frac{2-P_h^{k_{b,3}}}{2}\cdot\ldots, \qquad (7)$$

which also implies that for any *m* expected coverage close to *m=1*, all $k_{b,x}$ values must be sufficiently high.

As the broadcast reaches smaller and smaller subtrees, the number of messages will grow exponentially (as the number of subtrees is doubled in every step). With replication, the number of messages on an individual level is $2^{x-1}k_{b,x}$, where x is the height of the subtree and $k_{b,x}$ is the replication used on that level. (This formula is not applicable for the smallest subtrees, because the address space is not completely filled.) By increasing $k_{b,x}$ for subtrees of height *x*, we can enhance the coverage independently of the size of the subtree. This is easier in taller trees, where 2^{x-1} is much less than in smaller ones, but we must ensure proper delegation in smaller subtrees as well, so the possible reduction in network traffic is very small. Also, as discussed in the previous section and in Figure 6, the overall superfluous traffic created by replication of broadcast can sometimes be higher than expected, depending on the network latencies experienced by nodes.

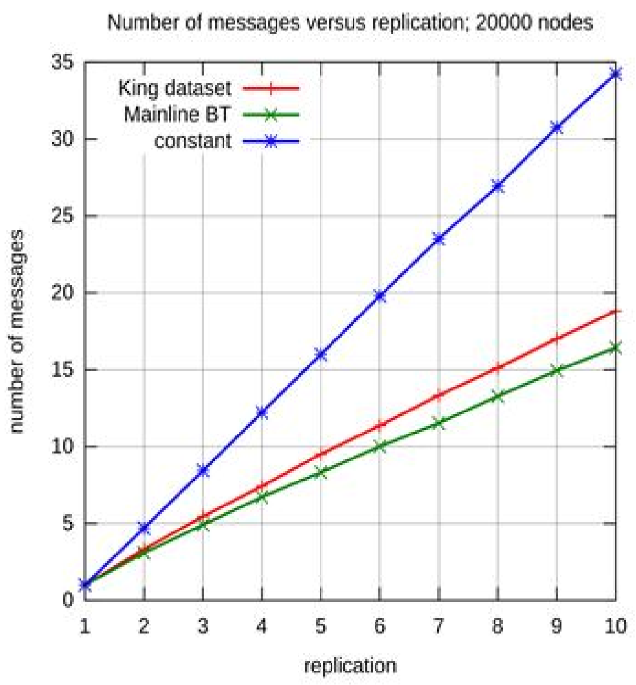
IV. Results

In order to evaluate the broadcast algorithms and to validate the model presented in Section III, we developed a specific simulator, which implements the necessary algorithms of Kademlia, and uses packet loss probabilities and latencies measured in real networks.

A. Reliability with Different Packet Loss Ratios

While reliability has a high deviation, on the average it can still be ensured by increasing replication. Figure 7 shows the simulation of an overlay with 1000 nodes. For a fairly high packet loss ratio P_h=20%, reliability can still be increased to 99% with only k_b=3.

Figure 7 also shows a comparison for the model presented and simulation. Blue lines indicate the model, while red dots show simulation results. Evidently, simulation shows a higher reliability than the one dictated by the model for k_b=3. This is caused by the inaccuracy of the estimation $P=1-P_h^{k,b}$ presented. The reliability gain of replication is only applicable for large subtrees. For small ones, in which there are only one or two neighbors, k_b=3 cannot be used. This would decrease overall reliability. However, when messages are duplicated, any node can receive the broadcast message through more than one path, some of which may be shorter than the path dictated by its Hamming distance. The average number of hops in which a node receives the message is lower, so the probability of getting it is higher. This has the effect that the reliability of the algorithm is higher than expected.



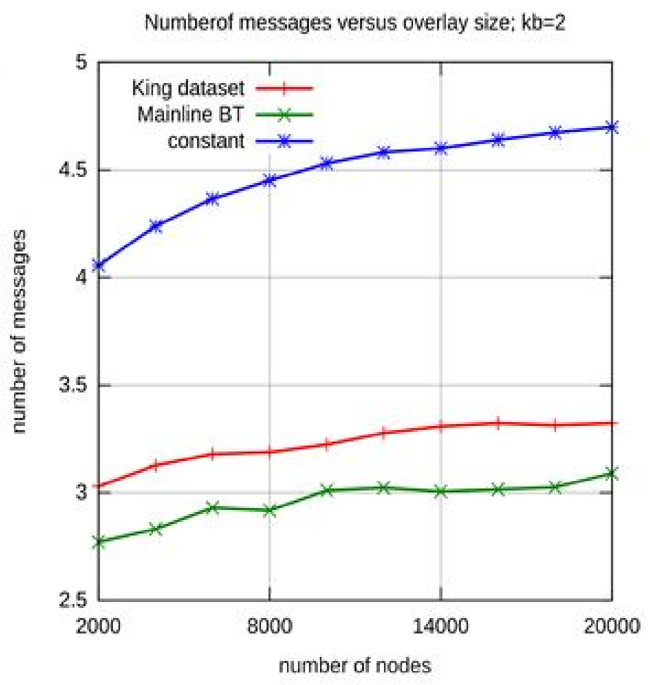


Figure 8. The number of messages per node of the broadcast algorithm for increasing replication and overlay size, when simulated with different network latencies between nodes

B. Number of Messages Required by the Broadcast

As discussed before, the algorithm presented in Section III uses *N*-1 messages only when parameterized not to use replication. When using replication the number of messages per node increases also with the level of replication and with the size of the overlay. Selecting multiple responsible nodes for subtrees has the side-effect that a single node can receive a broadcast message more than once. It is also possible that more network packets are in the network on the way to it at the same time. This phenomenon affects the number of messages generated by the algorithm. Consider the overlay of five nodes, with NodeIDs 0, 4, 5, 6 and 7 on Figure 6 for an illustration.

First assume that the broadcast is started by node 0, as seen on Figure 2. If k_b=2, it will send the message to two delegates in the distant subtree, 4 and 6, as shown by the black arrows. Both do the broadcast: node 4 sends the message to k_b=2 nodes, 6 and 7, and also to its neighbor, 5 (blue arrows). Node 6 acts similarly, by sending the message to 4, 5 and 7 (red arrows). Neglecting other messages still to be sent, this is 8 messages altogether.

If network latencies are diverse, the same broadcast can be carried out with totally different messages, as seen on Figure 6. Node 0 starts the message again, by sending it to 4 and 6. Now assume that to node 6 the network link has a very high latency. Node 4 receives the message much sooner, so starts the algorithm. In its distant subtree it can use replication (for 6 and 7), while in its closer subtree it cannot use replication, as node 5 is the only node in the whole subtree. Messages are quickly sent along the blue arrows and well. Now assume that the slow message (denoted by the red arrow) above now arrives. At this time, it will have no effect at all, because node 6 has already received he message from node 4, and discards the duplicate. The overall message count for this scenario is only 5.

This phenomenon can be observed in simulation results as well, as seen n Figure 8. When network latencies ere modeled after the "King" data set [13], more messages ere generated than in the simulated overlay using the latencies measured in the MainLine BitTorrent overlay [12]. The deviation of the values in the latter is much higher, so fewer messages are needed for broadcast. An overlay with totally uniform network latencies between any two nodes was also simulated. For that unreal boundary condition, much more messages were counted, which is consistent with the above discussion.

V. Conclusion

The novel algorithm presented in this article is applicable in the Kademlia overlay network to send quick and reliable broadcast messages. The algorithm uses no auxiliary routing tables other than those already created by Kademlia nodes; therefore it has zero additional maintenance costs. Nodes running the algorithm use multiple responsible node selection to enhance speed and reliability. As the model and simulation experience presented in the article indicate, replication allows one to increase reliability to any desired level at the cost of increased network traffic. However, nodes can calculate the required number of replicated messages in advance for an optimal trade-off between network cost and the reliability demanded by the application.

The article also investigated the possibility of using differentiated replication levels for each subtree size in the overlay for the broadcast algorithm. Although common sense dictates that replication is more important in subtrees with more nodes, both the model and the simulation results presented seem to contradict this. However, by having investigated the number of messages in each subtree, we found that replication is equally important in any of the subtrees, and the number of necessary responsible nodes is not dependent on the height of the subtree involved, once replication is estimated for the whole overlay. Our conclusion is that the same level of replication should be used for any subtree height, as the reliability gain of increasing replication in the higher subtrees only is marginal.

Acknowledgement

The work reported in the paper has been developed in the framework of the project "Talent care and cultivation in the scientific workshops of BME". This project is supported by the grant TÁMOP - 4.2.2.B-10/1-2010-0009.

References

[1] S. Androutsellis-Theotokis, D. Spinellis, A survey of peer-to-peer content distribution technologies, ACM Computing Surveys (CSUR) 36 (4) (2004) 335-371.

[2] P. Maymounkov, D. Mazieres, Kademlia: A peer-to-peer information system based on the xor metric, Peer-to-Peer Systems (2002) 53-65.

[3] J. Furness, M. Kolberg, A survey of blind search techniques in structured p2p networks, in: PGNet 2010: Proceedings of The 11th Annual PostGraduate Symposium on The Convergence of Telecommunications, Networking and Broadcasting, 2010, pp. 313-318.

[4] K. Huang, D. Zhang, A partition-based broadcast algorithm over dht for large-scale computing infrastructures, Advances in Grid and Pervasive Computing (2009) 422-433.

[5] J. Furness, M. Kolberg, Considering complex search techniques in dhts under churn, in: Consumer Communications and Networking Conference (CCNC), 2011 IEEE, IEEE, 2011, pp. 559-564.

[6] S. El-Ansary, L. Alima, P. Brand, S. Haridi, Efficient broadcast in structured P2P networks , Peer-to-Peer Systems II (2003) 304-314.

[7] S. Rhea, D. Geels, T. Roscoe, J. Kubiatowicz, Handling churn in a DHT, in: Proceedings of the annual conference on USENIX Annual Technical Conference, ATEC '04, USENIX Association, Berkeley, CA, USA, 2004, pp. 10-10.

[8] W. Li, S. Chen, P. Zhou, X. Li, Y. Li, An Efficient Broadcast Algorithm in Distributed Hash Table Under Churn, in: Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on, IEEE, pp. 1929-1932.

[9] P. Merz, K. Gorunova, Efficient broadcast in p2p grids, in: Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on, Vol. 1, IEEE, 2005, pp. 237-242.

[10] P. Trunfio, D. Talia, A. Ghodsi, S. Haridi, Implementing dynamic querying search in k-ary dht-based overlays, in: Grid Computing, Springer, 2008, pp. 275-286.

[11] V.M. Vishnevsky, A. Safonov, M. Yakimov, E. Shim, A.D. Gelman, Scalable blind search and broadcasting in peer-to-peer networks. In *Sixth IEEE Int. Conf. on Peer-to-Peer Computing*, 6-8 Sept. 2006, Cambridge, UK, 259-266.

[12] S. Crosby, D. Wallach, An Analysis of BitTorrent's Two Kademlia-based DHTs , Tech. Rep. TR-07-04, Department of Computer Science, Rice University (2007).

[13] K. Gummadi, S. Saroiu, S. Gribble, King: Estimating latency between arbitrary internet end hosts, in: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment, ACM, 2002, pp. 5-18.